Yale University
Department of Computer Science

Learning-Based Anomaly Detection in BGP Updates

Jian Zhang[1]
Computer Science Dept.
Yale University
zhang-jian@cs.yale.edu

Jennifer Rexford
Computer Science Dept.
Princeton University
jrex@cs.princeton.edu

Joan Feigenbaum[2]
Computer Science Dept.
Yale University
feigenbaum@cs.yale.edu

YALEU/DCS/TR-1318
April 2005

This work was supported by the DoD University Research Initiative (URI) administered by the Office of Naval Research.

[1]Supported by NSF grant number 0331548.
[2]Supported in part by NSF grants 0219018, 0331548, and 0428422 and by ONR grants N00014-01-1-0795 and N00014-04-1-0725.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
APR 2005		**00-04-2005 to 00-04-2005**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Learning-Based Anomaly Detection in BGP Updates	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Yale University,Department of Computer Science,PO Box 208285,New Haven,CT,06520-8285**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**12**	

Learning-Based Anomaly Detection in BGP Updates

Jian Zhang*
Computer Science Dept.
Yale University
zhang-jian@cs.yale.edu

Jennifer Rexford
Computer Science Dept.
Princeton University
jrex@cs.princeton.edu

Joan Feigenbaum†
Computer Science Dept.
Yale University
feigenbaum@cs.yale.edu

Abstract

We propose an instance-learning based framework for detecting BGP routing anomalies. By using a vector of quantified features to represent BGP updates, our framework can capture more complex features of BGP updates than previous methods that use simple aggregation. The feature vector is based on BGP-update dynamics and is constructed using wavelet transformations. The transformations provide a systematic, multi-scaled analysis of the dynamics and thus avoid using "magic numbers" that are hard to determine.

We experiment with a preliminary implementation of our framework, investigating daily BGP update behaviors for six months. Focusing on each prefix in isolation, we show that, for most prefixes, update dynamics are similar from day to day. Furthermore, on a single day, most prefixes also display similar dynamics. Only a few prefixes exhibit behaviors that are quite different from the majority. The small set of prefixes or daily behaviors can be further examined for anomaly detection. In particular, we observe that most prefixes whose update dynamics deviate from the majority are unstable prefixes with frequent routing changes.

Keywords: BGP Anomaly Detection, Wavelet Transformation

This work was supported by the DoD University Research Initiative (URI) administered by the Office of Naval Research.

*Supported by NSF grant number 0331548.

†Supported in part by NSF grants 0219018, 0331548, and 0428422 and by ONR grants N00014-01-1-0795 and N00014-04-1-0725.

1 Introduction

The stability of BGP affects the stability, availability, and efficiency of the Internet. It is thus of great importance to understand the behavior of BGP. In recent years, a lot of work has been done along these lines [2, 3, 5, 8, 10, 12]. Some studies view any route change as a sign of instability. However, not all route changes are "bad," in that not all cause instability problems for the network. In some cases, route updates may be normal responses by BGP to network changes. In other cases, they may be the result of traffic-engineering efforts. These route changes actually can help to improve the performance of the network. It is only "abnormal" route changes (*e.g.*, frequent updates due to flaky equipment, protocol oscillation, route hijacking, etc.) that require the network operator's attention.

Of course, the notion of "abnormal" route changes ("abnormal" BGP updates) is not well defined. One AS's "abnormal" BGP updates may be another AS's "normal" updates. The definition depends on different AS's configurations and policies. Although a general definition may be hard to obtain, it is helpful to consider features/attributes of BGP updates that may be used to identify "anomalies." Ideally one would also like to automatically learn the "normal" behaviors and use what has been learned to distinguish the "abnormal" from the "normal."

Previously, statistics-based anomaly detection[11, 14] has been used in this context. A statistics-based detection system [6] attempts to detect anomalies by comparing the current behavior with the history of known behaviors. If the current behavior deviates a lot from the known history, it will be flagged as a possible anomaly. In a statistics-based system for detecting BGP-route anomalies, the behaviors of BGP updates are normally measured and represented by simple aggregates such as the number of updates within a certain period of time or the time it takes for a prefix to converge. The history consists a simple statistic (the mean and the variance) of these aggregates, and traditional statistical tests can be used to see whether the current behavior deviates significantly from the normal behaviors.

The statistics-based approach is simple and thus may be easily deployed and run with high efficiency. This makes it attractive in situations where a huge amount of data need to be processed. However, the very simplicity of this representation also makes it unable to capture complex features that may be important for a better analysis of BGP behavior. Furthermore, the representations in many such systems have the "magic-number" problem. That is, they use parameters, set either arbitrarily or according to statistics, for controlling granuality of the analysis or for the threshold that determines when a burst of messages ends. For example, a prefix may be determined to have converged to a stable route if there have been no updates for that prefix for T minutes. Clearly, it is hard to set a good value for such a parameter T (although it is easy to give a very loose upper bound for T, *e.g.* T should be smaller than 50,000.)

These problems motivate our search for new representations and new frameworks that are independent of "magic numbers" and more powerful in characterizing BGP updates. At the same time, the framework should still be efficient and deployable. Towards this end, we propose a framework using instance-based learning. Instance-based learning is a form of nearest-neighbor learning, where the label (say "normal" or "abnormal") of a behavior is determined by the label of the most similar behavior in the knowledge base. Hence, at a high level, our learning-based anomaly detection bears a resemblance to the statistics-based anomaly detection. Both approaches detect abnormal behavior by comparing it to a history of known behaviors. At the detailed level, however, the two

approaches differ greatly in how the behaviors are represented, how the history is compiled, and how the current behavior is tested against the history.

In our framework, BGP update behaviors are represented by a vector of quantified features. Such a representation maps a particular behavior to a point in a multidimensional vector space. The set of normal behaviors map to a set of points whose neighborhood defines the domain of "good" behaviors. A behavior represented by an outlier point that is far away from this domain would be suspicious and may require the network operator's attention. The domain of normal behaviors and the outlier points can be discovered by clustering. Thus, in our framework, the clusters of known behaviors form a knowledge base. To check the current behavior against the history is to perform a nearest-neighbor query in the knowledge base. By extending the representation to a vector of quantified features, our framework is much more expressive in its ability to characterize BGP-update behaviors.

In this preliminary study, we use the features of BGP-update dynamics to construct the representation vectors. That is, we look for features in temporal patterns of the BGP-update dynamics, such as burst duration, inter-burst intervals, *etc.* We argue that features in BGP-update dynamics are relevant to the detection of BGP-update anomalies. We will elaborate on this argument in next section when we describe our representation in detail. To extract the features (temporal patterns) of BGP-update dynamics, we use wavelet transformations. In signal processing, wavelet transformations map a raw signal (a function of time) into a signal (or coefficients) of the time-frequency domain (a function of both time and frequency). Note that, if we treat a sequence of update messages as a signal along time, *i.e.*, a function $f(t)$ whose value is the number of updates (for a view or for a particular prefix) at time t, an update-message burst within this signal can be viewed as a high-frequency signal (the individual updates) modulated by a low-frequency signal (the burst). The wavelet transformation can thus reveal the temporal structures in the update-message dynamics.

Our representation has several advantages. First, it avoids the "magic-number" problem by employing a multi-scale transformation. Using a set of different frequencies, the wavelet transformation provides a systematic, multi-granuality view of the structures and patterns of BGP updates. Note that we view the update dynamics at different levels: 1 second, 2 seconds, $\cdots \ell_{max}$ seconds. ℓ_{max} determines the upper bound on the scales of our view. As discussed above, for many parameters, although it is hard to determine their exact values, a loose upper bound can always be obtained. (There is a reasonable range for these parameters. They can not be arbitrary values in $(-\infty, +\infty)$.) We can set ℓ_{max} to be such a upper bound and investigate the set of values in the range in a systematic way. Therefore, our system avoids the "magic-number" problem not by magically getting rid of all the parameters but by systematically examining the intervals in which the parameters may lie.

Another advantage of our representation is its shift-invariant property. Our anomaly-detection framework compares the current BGP-update dynamics with the dynamics in the knowledge base. Two sequences of update dynamics may be very alike, but they may not be well aligned in time. For example, if the normal behavior of a particular prefix is to undergo a short burst of updates once in a while, it shouldn't matter if the updates happen at 12:00 or 12:30. However, a shift-intolerant representation would view the two sequences of dynamics as quite different behaviors and misclassify them. Furthermore, update dynamics of different prefixes are normally not well aligned in time. Even the updates of multiple prefixes caused by a single event may shift in time

because of rate limiting by protocol timers or by the transmission protocol. Again, a shift-intolerant representation will characterize them as very different behaviors. Classification according to such a representation will not produce meaningful categories of BGP-update dynamics. Because our representation is shift-invariant, it provides a metric space in which comparison and classification don't have these problems.

Because of the intrinsic complexity of BGP-update anomaly detection, humans (network operators) retain a critical role in our anomaly detection framework. The goal of our framework is not to eliminate human investigation. Rather, the framework tries to facilitate human investigation by selecting a small set of anomaly candidates. It is the network operators who make the final decision and take action.

We used a preliminary implementation of our framework to investigate BGP-update behaviors for 6 months. Our results shows that, for a single prefix, update dynamics are quite consistent. That is, most of the prefix's update dynamics are similar. Only a few update dynamics are different from the majority. For a set of prefixes over the same view, the same phenomenon is exhibited. Thus our system distinguishes a small set of possible abnormal update dynamics that can be scrutinized further by network operators.

We remark that this is a preliminary study. Certain implementation decisions have not been fully tested by the data. Thus, some patterns of BGP update dynamics may not be exposed by this particular implementation. Secondly, although the framework we proposed is quite general, the particular representation we used in this study relies solely on the update dynamics for anomaly detection. The representation does not include other information in the update message, such as the path of the newly announced routes. We intend to include more such information in an extended representation in future work. Finally, another important piece of future work is to validate, via data from ISPs and known Internet outages/anomalies, whether the anomalies detected by our system are truly "anomalous."

2 Anomaly Detection Framework

In this section, we present in detail our framework for detecting BGP-update anomalies. First, we argue that focusing on BGP-update dynamics is a good approach. Then we describe how the update dynamics are represented in our framework. Finally, we show how clustering is used to mine the data and to improve the efficiency of the system.

2.1 Feature Extraction in BGP-Update Dynamics

The reason for focusing on BGP-update dynamics is the following: As discovered in [7], different update types have different convergence time and number of update messages. The convergence and update-message numbers also differ from ISP to ISP. Hence, the configuration of the underlying network affects the way the new paths are explored, which in turn can lead to different timing and message numbers in different systems. In fact, the work in [4] used BGP-update dynamics to infer the network's topological information. If we assume that the underlying network does not change configuration often, the temporal dynamics of the updates should also be similar. Thus, unusual dynamics may indicate anomalous updates, as the analysis of BGP updates during certain

worm attacks have shown [9]. Indeed, several recent works [14, 13] have examined the BGP-update dynamics for signs of anomaly.

We extract the features of BGP-update dynamics using wavelet transformations. We now describe the transformations and the construction of the representation vectors. We use BGP-update data from RouteViews [1]. Recall that we avoid the "magic-number" problem not by getting rid of all the parameters but by systematically examining the interval in which the possible values of the parameters fall. In this preliminary study, we examine the interval upper-bounded by 24 hours (this upper bound can be greatly extended in a full scale system). That is, we take the daily BGP updates as a unit (an episode) of a single prefix's update behavior. The daily updates are viewed as a sequence. The i-th entry in the sequence is the number of updates at time i (i-th second). We denote by S this sequence and by n the length of the sequence. We perform a (discretized) continuous wavelet transform on this sequence. Because of their simplicity, we use Haar wavelets in this preliminary implementation. We intend to experiment with other forms of wavelets in a full-scale study. Let $\Psi(\frac{x-\tau}{\delta})$ be the Haar wavelet with translation τ and scale δ. The discretized transformation is defined to be $\gamma(\delta,\tau) = \sum_x S(x) \cdot \frac{1}{\sqrt{\delta}}\Psi^*(\frac{x-\tau}{\delta})$. The transformation employs a set of scales $(\delta_0, \delta_1, \ldots \delta_\lambda)$. We start with a wavelet of scale of 20 seconds. That is, $\delta_0 = 20$. The other scales have the form $\delta_{i+1} = 2\delta_i$.

In the continuous transformation, τ takes values from the set $\{1, 2, \ldots, n\}$. The result of the transformation is a function $\gamma(\delta,\tau)$ of both time (τ) and frequency ($1/\delta$). The transformation itself has a size larger than the original sequence and is unsuitable to be used as a succinct representation directly. However, we are only interested in the peak values of $\gamma(\delta,\tau)$. This is because a burst of length t will give a large value of $\gamma(\delta,\tau)$ at the scale δ most close to t. Thus our representation consists of only the peak values of $\gamma(\delta,\tau)$ and the time intervals between each pair of consecutive peaks.

To further reduce the size of the representation, instead of recording the exact peak values of $\gamma(\delta,\tau)$, we only keep an approximate value. Let γ_{max} be the largest peak value of $\gamma(\delta,\tau)$. We divide the interval $(0, \gamma_{max}]$ into bins of the form $(0, \nu]$ and $(\nu \cdot (1+\epsilon)^i, \nu(1+\epsilon)^{i+1}]$, $i = 0, 1, 2 \ldots$. (In this experiment, we set $\nu = 0.1$ and $\epsilon = 0.5$) A histogram of the peak values of $\gamma(\delta,\tau)$ can be built using these bins. Note that such a histogram is a vector V_γ whose i-th entry represents the number of the peak values of $\gamma(\delta,\tau)$ that fall into the corresponding bin. A similar histogram (vector V_i) can be constructed for the inter-peak intervals. The combination of V_γ and V_i gives the final representation: a vector that is more complex than a single aggregation but still has size much smaller than the original sequences.

2.2 Grouping Dynamics Using Clustering

In our framework, clustering serves two purposes. First, clustering is used to mine and discover the structures in the data. The points representing the update dynamics may not be uniformly distributed over the vector space. Rather, many of them may be close to each other while some of them may be outliers. Clustering helps to discover such structures. Note that such a structure provides important information for detecting anomalies. Assuming that the majority of the BGP-update behaviors are normal, the large clusters that contain many points automatically define the normal behaviors and very small clusters that contain only few points can form a good candidate set for anomaly. Secondly, clustering helps to improve the efficiency of the system. Instead of

populating our knowledge base with thousands of points, we cluster them and store the clusters (the center and the radius) in the knowledge base. This reduces storage space and computing time. Finally, our system still requires a network operator to recognize abnormal behaviors. With clustering, the operator need only scrutinize a small number of clusters instead of going through many behaviors that are similar.

In this study, we considered BGP updates for 6 months, from June 2004 to November 2004. For each month, we use the data of the first 30 days. Hence, each prefix has 180 episodes of update behaviors in our knowledge base. Each of the episodes is represented by a vector computed using the aforementioned method. We experiment with two types of clusterings on the vectors. First for each prefix, we perform clustering on its 180 episodes of update dynamics. Second, we perform clustering across prefixes over a view on a randomly chosen day. k-means clustering is used for both type of clusterings. In k-means clustering, k, the number of clusters, is a predefined parameter. In our framework, it controls the trade-off between the sensitivity of the detection and the efficiency of the system. When applied on a set of m points, the clustering method produces k clusters, $C_1, C_2, \ldots, C_k$, of points and k centroids, $u_1, u_2, \ldots, u_k$, of the clusters. The sum of the distances between each point and its corresponding centroid is minimized, *i.e.*, the clustering method minimizes $\sum_{i=1}^{k} \sum_{p \in C_i} d(p, u_i)$, where $d(p, q)$ is the distance between points p and q. In our experiments, the distance between the points in the vector space is defined to be their Euclidean distance.

In the sections follow, we present some of the structure of BGP-update dynamics revealed by our preliminary experiment and show how they may help to detect BGP-update anomalies.

3 Dynamics for a Single Prefix

In this section, we examine the results of the first type of clustering. Here we cluster the 180 vectors belonging to the same prefix. Recall that our framework builds on a basic idea that, for most prefixes, their history consists mostly of normal behaviors. The anomalies always consist of the very few behaviors that deviate from the majority. Thus we expect skew in the size of the clusters, *i.e.*, one or two clusters may contain most of the 180 points, while the rest only have a few points in them. Our clustering results show that this is the case.

Figure 1 shows the typical profile of the size of the clusters for a prefix. Note that we set the number of centers to be 10 in this clustering. Each bar in the figure corresponds to a cluster. The height of the bar indicates the number of points in the cluster. We can see that the number of points drops dramatically from the largest cluster to the second largest cluster. The largest cluster contains over 90 percent of the 180 points, while most others contain only several points. We observe that, for many prefixes, the dynamics in the small clusters often have more update bursts than the dynamics in the large cluster. The duration of the bursts and the number of messages within a burst may slightly increase too.

To gather additional evidence that cluster sizes have the profile in Figure 1, we randomly choose 200 prefixes and show the average size of the clusters. For each of the 200 prefixes, we do clustering on the 180 points representing its update dynamics. The resulting clusters are sorted according to their size. Now we have 200 sets of clusters. Within each set, we have 10 clusters, sorted in decreasing order by size. The size of the overall largest cluster is then determined by the average of

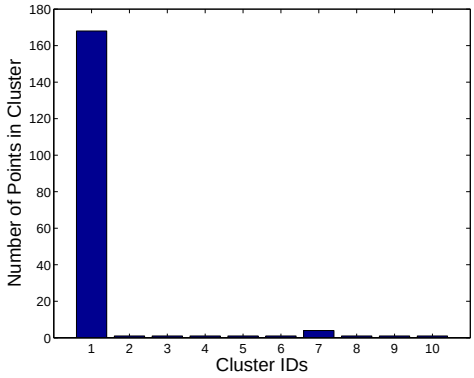

Figure 1: Cluster Sizes for a Single Prefix's Update Dynamics

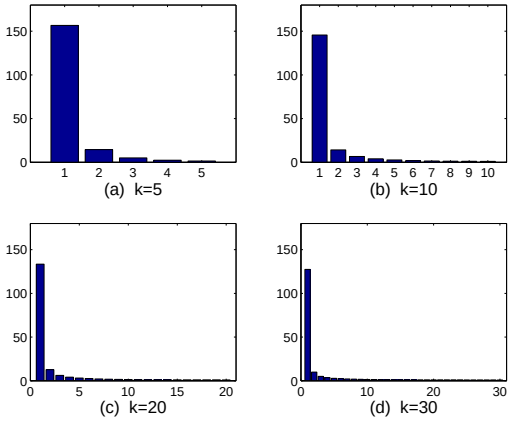

Figure 2: Average Cluster Sizes and the Effect of k

the size of the 200 largest clusters in each set. The size of the second largest cluster is the average of the size of the 200 second largest clusters in each set, *etc.*

Panel (b) in Figure 2 shows the average size of the clusters. Clearly, the largest cluster still dominates over all the clusters. It validates the idea that the BGP-update dynamics of a single prefix is quite consistent and that this can be used as the basis for anomaly detection. We remark that any detection system based on learning needs to see sufficiently large number of previous behaviors to give a good classification. As a preliminary study, we used a short history (6 months). Thus, being clustered into a small cluster in this particular case does not necessarily mean that the behavior is "bad." But, with a longer history, one can expect the difference between the number of normal and abnormal behaviors to increase.

Our clustering method requires a parameter, the number of clusters k. With a large k, the update dynamics will be classified into more categories, which enables more detailed comparison between the dynamics. On the other hand, a system with small k will require less storage space and computing time. To see how k affects our clustering, we examine the average cluster sizes under different choice of k. The result is plotted in Figure 2. Panels (a), (b), (c), and (d) correspond to

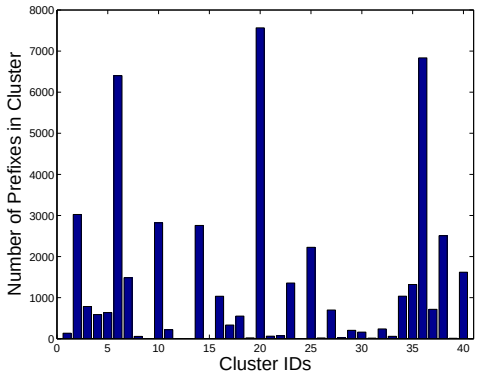

Figure 3: Cluster Sizes when Clustering across Prefixes

the profiles of the average cluster sizes when k is set to 5, 10, 20, and 30, respectively. Note that, in all cases, the profile remains similar. As the number of clusters increases, the sizes of the large clusters decreases slightly. But the largest cluster still remains dominant.

4 Dynamics across Prefixes

In addition to one prefix's own history of update dynamics, comparing one prefix's update dynamics with other's can also aid in anomaly detection. In this section, we discuss the phenomena revealed by the second type of clustering. Here we cluster across prefixes. There are tens of thousands of prefixes that undergo updates in a single day. The dynamics of each prefix map to a point in our vector space. Clustering groups together the prefixes that display similar dynamics. (The clusters in this section consist of prefixes while the clusters in the previous section consist of episodes of dynamics belonging to the same prefix.)

We first show that dynamics of most prefixes are quite similar. Figure 3 shows the clusters across different prefixes for Aug. 8th. Each bar corresponds to a cluster, and the height of the bar indicates the number of prefixes in that cluster. The majority of the prefixes are contained in 3 clusters (number 6, 20 and 36). A close examination shows that these are the prefixes that only display very few events(bursts) of updates. Samples also show that the prefixes in cluster 20 only have one or two short bursts and several sporadic updates from time to time during the day, while the prefixes in cluster 6 have several short bursts and almost no sporadic updates. We also observe that there are clusters that contain merely 1 or 2 prefixes. Most of these tiny clusters contain prefixes that are constantly being updated on that day. The difference between a prefix in a large cluster and a prefix in a small cluster can also be seen when we cluster their update dynamics using the first type of clustering. For most prefixes in the large clusters, the results show a profile like the one in Figure 1. For many prefixes in the small clusters, the cluster sizes derived by the first type of clustering are less skewed. Thus the prefixes in the small clusters have more complex behaviors, which agrees with the fact that they often have more updates.

Another factor that divides the prefixes into different clusters is the correlation between their update messages. When updates for two prefixes are correlated, they display almost the same

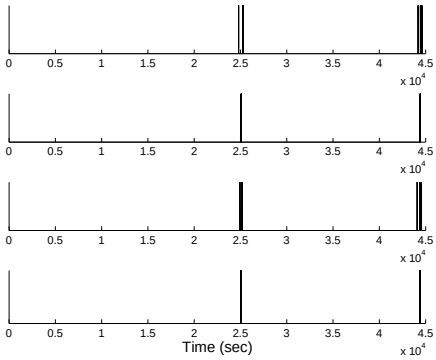

Figure 4: Correlated BGP Update Messages Discovered by Clustering

temporal patterns and thus are more likely to be included in the same cluster. On the other hand, non-correlated prefixes are more likely to split into different clusters. In Figure 4, we plot the updates of four prefixes in cluster 20 of Figure 3. The plot shows the raw updates of the 4 prefixes where each spike indicates an update message for the corresponding prefix. The messages are highly correlated, and one may speculate that the updates of these prefixes are caused by the same network event.

Clustering across prefixes also helps to identify prefixes with "bad" behaviors. If we know in advance that some "troublesome" prefixes exhibit "bad" behaviors such as constant change, we can expect that other prefixes in the same cluster will behave similarly.

In Figure 5, we plot the behaviors of two prefixes that are both in cluster 22 of Figure 3. For each prefix, daily update dynamics are represented by a vector of dimensionality 200. In the plot, we use gray scale to indicate the value of the entries in the vector. The darker the color, the larger the value of the corresponding vector entry. The x-axis is time. The column of gray-scale pattern at a certain x (say x = 60) is our representation for the prefix's update dynamics on that day (say July 30th). Note that the prefix plotted in the upper panel has a lot of activity in our representation, because it is constantly changing. The underlying problem that causes this instability was fixed in Sept and the prefix became stable. Because we cluster across the prefixes for the day of Aug. 8th, the prefix plotted in the lower panel falls in the same cluster as the one plotted in the upper panel. We can see that, although the second prefix is stable most of the time, it does exhibit similar changing behavior for two days during the 6-months period. One can expect that any behavior that maps to a point close to these two will be suspicious.

5 Related Work

There has been many research work on analyzing BGP instabilities [2, 3, 5, 8, 12]. In [14], Zhang *et al.* combine signature-based and statistics-base approaches to detect BGP-route anomalies. They use NIDES [6], a system originally developed for intrusion detection, as the statistics-based component. They show that the system using combined approaches can identify certain BGP events that are worthy further investigation.

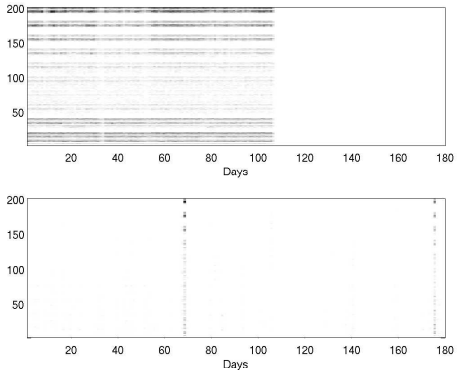

Figure 5: Abnormal Dynamics Discovered by Clustering

Xu *et al.* [13] infer BGP events by PCA (principal component analysis). In their work, one AS's BGP dynamics are represented by a vector whose entries consist of the numbers of updates in intervals of 30-seconds. PCA discovers the set of principal components *i.e.*, the intervals of time when the ASes display the most variances in their updates. The authors show that such information can be used to infer major BGP events. Our idea shares a similarity to theirs in that we both view variances in the dynamics as a sign of events. Our framework differs from theirs by employing a multi-scaled analysis and by including more features of BGP-update dynamics in the representation. We also investigate the dynamics in the signals where there are less variances. We use features from these signals to model the normal BGP-update dynamics.

6 Summary and Future Work

In this paper, we propose an instance-learning based framework for detecting BGP-routing anomalies. We consider BGP-update dynamics in this framework. We also propose a wavelet-transformation-based vector representation for BGP-update dynamics. We use clustering to mine the data and improve the system efficiency. Our preliminary results show that the update dynamics of most prefixes, most of the time, are consistent, *i.e.*, their dynamics are similar. Only a few prefixes exhibit behaviors that are quite different from the majority. This small set of prefixes may be further examined for anomalies. Furthermore, the results of clustering may provide information helpful for locating correlations between prefixes and for characterizing anomalies.

As a preliminary study, the system given in this paper lacks sophistication and fine tuning. We propose the following future goals for a more comprehensive evaluation and understanding of the framework:

1. We plan to explore the "abnormal prefixes" and the "abnormal clusters" (for the same prefix) to understand the relationship between the structures and patterns revealed by our system and the operation of BGP. Furthermore, we plan to validate, with other sources of data, whether the "anomalies" (the outliers) discovered by our system are truly "anomalous."

2. Our current representation relies only on update dynamics to detect anomalies. It would be helpful to include more information (such as newly announced paths) in the representation.

3. In a short period of time, the normal update dynamics of a prefix may remain similar. With the evolution of the network, the features in update dynamics may change over time. It may be necessary to add data-aging mechanism that adapts to more recent features.

References

[1] Route Views Project. `http://www.routeviews.org`.

[2] M. Caesar, L. Subramanian, and R. H. Katz. Towards localizing root causes of BGP dynamics. In *Technical Report, CSD-3-1292, UC Berkeley*, 2003.

[3] D.-F. Chang, R. Govindan, and J. Heidemann. The temporal and topological characteristics of BGP path changes. In *Proc. IEEE ICNP*, 2003.

[4] G. A. David, N. Feamster, S. Bauer, and H. Balakrishnan. Topology inference from BGP routing dynamics. In *Proc. IMW*, 2002.

[5] A. Feldmann, O. Maennel, Z. Mao, A. Berger, and B. Maggs. Locating internet routing instabilities. In *Proc. ACM SIGCOMM*, 2004.

[6] H. Javitz and A. Valdes. The NIDES statistical components: Description and justification. In *Technical report, SRI Network Information Center*. 1993.

[7] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. *IEEE/ACM transactions on networking*, 9(3):293–306, 2001.

[8] M. Lad, A. Nanavati, D. Massey, and L. Zhang. An algorithmic approach to identifying link failures. In *Proc. Pacific Rim Dependable Computing*, 2004.

[9] M. Lad, X. Zhao, B. Zhang, D. Massey, and L. Zhang. Analysis of BGP update surge during slammer worm attach. In *Proc. 5th International Workshop on Distributed Computing*, 2003.

[10] J. RexFord, J. Wang, Z. Xiao, and Y. Zhang. BGP routing stability of popular destinations. In *Proc. IMW*, 2002.

[11] S. T. Teoh, K. Zhang, S.-M. Tseng, K.-L. Ma, and S. F. Wu. Combining visual and automated data mining for near-real-time anomaly detection and analysis in BGP. In *Proc. VizSEC/DMSEC*, 2004.

[12] J. Wu, Z. M. Mao, J. Rexford, and J. Wang. Troubleshooting BGP disruptions in a large IP network. In *Proc. Networked Systems Design and Implementation*, 2005.

[13] K. Xu, J. Chandrashekar, and Z. Zhang. Inferring major events from BGP update streams. In *Technical Report, Dept. Computer Science and Engineering, Univ. Minnesota*, 2004.

[14] K. Zhang, A. Yen, X. Zhao, D. Massey, S. F. Wu, and L. Zhang. On detection of anomalous routing dynamics in BGP. In *Proc. NETWORKING LNCS 3042*, pages 259–270, 2004.